Author(s): Jelovica, Jasmin & Klanac, Alan

Title: Healing and repairing techniques for faster optimization with genetic algorithm

Year: 2009

Version: post print

Please cite the original version:

Jelovica, Jasmin & Klanac, Alan. 2009. Healing and repairing techniques for faster optimization with genetic algorithm. Proceedings of the 10th Finnish Mechanics Days (Suomen mekaniikan päivät 2009).

‘HEALING’ AND ‘REPAIRING’ TECHNIQUES FOR FASTER OPTIMIZATION WITH GENETIC ALGORITHM

J. JELOVICA & A. KLANAC

Marine Technology, Department of Applied Mechanics

Teknillinen korkeakoulu

P.O.Box 5300

FIN-02015 TKK, FINLAND

jasmin.jelovica@tkk.fi; alan.klanac@tkk.fi

ABSTRACT

This paper presents two techniques, the ‘healing’ and ‘repairing’ that can reduce optimization time when using genetic algorithm for structural optimization. The techniques can be applied to: (a) quickly find feasible designs from completely infeasible set of alternatives, and (b) to make the best infeasible designs feasible. These procedures are implemented into a genetic algorithm ‘VOP’. The performance of the original and the modified version of the algorithm are compared with the widespread genetic algorithm ‘NSGA-II’ for the weight optimization of a 40 000 DWT chemical tanker midship section. The results show that these procedures can decrease the optimization time by approximately half.

1 INTRODUCTION

Design of modern ships introduces new complex structural solutions that must follow the increasing demand for more reliable and safe products. However, the available time does not follow the increasing complexity of design procedure, thus more advanced support systems are required that can assist designers. Such a system could be based on the optimization process. The intention of this study is to show how to enhance this process with respect to the increase of speed.

Complex ship structures involve large number of variables and even larger number of constraints. Variables are in structural optimization regularly discrete, whether they represent element size, material type, stiffener spacing etc. Constraints are non-linear and non-convex, typically involving yielding and buckling of structural elements. These reasons confine the choice of possible optimization algorithms to those that do not require gradient calculation of constraints and objective functions. Evolutionary algorithms have shown capability to handle such problems and provide sufficient benefits for the structure. One of them is the genetic algorithm (GA). Several applications have shown that GA can be a successful tool for practical problems in ship structural design and optimization, see e.g. Romanoff and Klanac (2007), Ehlers et al. (2007), Klanac and Jelovica (2009), Klanac *et al.* (2009).

Genetic algorithm operates in the design space, by having multiple design alternatives at hand when deciding where to continue the search from generation to generation. This number of available solutions is known as a population size and should grow with the number of considered

variables. Literature suggests using population size in range from 50 to 500; see Deb (2001). This lengthens the optimization time even for the simple engineering problem, since the number of generated and evaluated design alternatives before reaching the optima can be more than several thousands. Clearly, this can be rather costly when optimizing large ship structures, especially if *Finite Element Method* is applied for structural assessment. In any case, optimization should be short, and if it is time-consuming, it is often, for convenience, stopped prematurely, immediately after noticing some improvements in objective values and without attaining their optima. Making relevant conclusions based on such results can be misleading and costly in the later stages.

Optimization in tightly-constrained design space typically involves infeasible designs that do not satisfy all requirements included in the problem. Such designs are considered less valuable than the feasible designs, and are suppressed or completely neglected as for example in NSGA-II (Deb et al. 2002) or ε-MOEA (Deb et al. 2003) genetic algorithms. Here however, a different approach is suggested. Instead of rejection, infeasible design can be *repaired* so that it hopefully yields feasible design, partially maintaining the original's good objective(s) value(s).

The two procedures that form this approach are implemented in a GA called VOP. The original and the modified version of the algorithm are compared with NSGA-II, a recognized genetic algorithm that possesses several advanced features. To demonstrate this comparison, a structure of a 40 000 DWT and 180m long chemical tanker is optimized for the two objectives: minimum of hull steel weight and minimum of duplex steel weight.

The following chapter describes heal and repair techniques and the way how they can be implemented in the GA. Chapter 3 describes the VOP algorithm. Chapter 4 presents the test case that is used for optimization in Chapter 5, where heal and repair procedures are utilized and validated with NSGA-II. The last chapter concludes the paper.

2 HEALING AND REPAIRING FOR FASTER OPTIMIZATION

Diverse set of designs in each generation allows the GA to select better-than-average individuals that will be used for optimization continuation. Infeasible designs are in that sense considered less valuable and completely rejected in the prominent algorithms like NSGA-II or ε-MOEA once the feasible space has been found. But oftentimes those infeasible designs are much better according to objective(s) values(s). In the case where objective(s) depend on the numerous designs variables, for example in the weight optimization of a complex structure, changing few of them would not significantly alter the performance of a design. Then it could be useful to alter such design as little as possible to make it feasible but to retain its objectives.

Various schemes have been reported on how to make infeasible designs feasible. Oftentimes they are based on designer's insights on the physical problem they are dealing with. Certain way is observed how to satisfy a requirement and the designs are composed to fulfill it; see for example Todoroki and Haftka (1998), Liu et al. (2000), Chou et al. (2001), Cheong and Lai (2000). Repair technique is conceived here in a more generic manner. Designs are not improved based on some prior-knowledge specific to a problem at hand, instead they benefit from other solutions present in the current generation. However, there is one condition: a sensitivity of the constraints to variables should be known, $x_l \rightarrow g_j$, so that the variables that caused constraint violation can be identified.

This variable-constraint 'link' should be relatively easy to determine in optimization problem since constraints are always functions of decision variables, $g=g(\mathbf{x})$. At this point, the term 'infeasible' is extended beyond description of a particular design, identifying now a specific constraint that is violated, $g_j < 0$, and associated variable x_l. If a design is infeasible, there exists at least one broken constraint and corresponding infeasible variable. The idea is then to replace that infeasible variable with the variable from different design that does not violate the same constraint.

The proposed approach is divided in two procedures that can be applied in the following cases: (a) to quickly find feasible solutions from completely infeasible population and (b) to make the best infeasible designs feasible, in the presence of feasible designs in the population. The former is named 'heal', and the later 'repair' procedure. Both are described in the sequel.

2.1 Healing

If all designs in the population are infeasible, it does not mean that all of their constraints are too. When a particular constraint is observed, there exist designs in the population which do and do not break it, or even if they all break it, some do it less than the others. To make a design feasible, its infeasible variables are identified and replaced with corresponding feasible variables from a design that is best in objective(s) value(s). Besides 'healing' the design, i.e. turning it into feasible, the intention is to make it competitive objective-wise. There are two interesting categories of designs in the population: those with the smallest overall constraints violation and those with the best objective(s) value(s). The former offers a grater probability to become feasible and requires fewer variables to change. The latter group is tempting since designs with good objective(s) value(s) are the goal of the optimization, but are likely to have more infeasible variables that should be healed.replaced. It would be quite useful to make them feasible and simultaneously not significantly deteriorate their objective value(s). Oftentimes these two features are contrary in structural optimization where weight minimization makes light designs violate many strength constraints, while the feasible ones can in the beginning be quite heavy.

The suggested strategy is to heal N_{CON} of the best designs that violate the least amount of constraints and also N_{OBJ} of the best designs according to objective(s) value(s), starting with the best design in each category. Regardless of the category, every design which undergoes healing process gets its infeasible variables replaced with a feasible value from the best performing design for that variable.

To sort design alternatives according to their constraint violation, the following expression is used:

$$S_{CON}(\boldsymbol{x}) = \sum_{j=1}^{J} \mathrm{H}\left[g_j(\boldsymbol{x}) \right] \tag{1}$$

where J is the number of constraints in the problem and $\mathrm{H}[\cdot]$ is a Heaviside operator defined by

$$\mathrm{H}\left[g_j(\boldsymbol{x}) \right] = \begin{cases} -g_j(\boldsymbol{x}), \text{if } \mathrm{g}_j(\boldsymbol{x}) < 0 \\ 0, \qquad \text{otherwise.} \end{cases} \tag{2}$$

Before using the Equation (1) infeasible constraint values are linearly normalized between 0 and 1.

The performance of the design in the normalized objective space is measured by

$$d_{OBJ}(\boldsymbol{x}) = \left\{ \sum_{m} \left[1 - f_m(\boldsymbol{x}) \right]^2 \right\}^{1/2}. \tag{3}$$

The heal procedure is described below:

Step 1: Rank the population according to the smallest overall constraint violation using Eq. (1).
Step 2: Rank the population according to the best objective(s) value(s) using Eq. (3).
Step 3: Determine infeasible variables to the best N_{CON} designs that are ranked based on S_{CON} and the best N_{OBJ} designs based on d_{OBJ}.
Step 4: Starting from the best S_{CON} design, do for each N_{CON}: keep all feasible variables and replace each infeasible with a feasible value from best possible design according to d_{OBJ}.

If there is no feasible value from a particular variable in the population, take the one from the best S_{CON} design.

Step 5: Starting from the best d_{OBJ} design, do for each N_{OBJ} : repeat the action from the *Step 4*.

Population size must remain the same, thus $N_{CON} + N_{OBJ} \leq N$. Beside the designs selected for healing, remaining designs are chosen in a standard way, which can differ depending upon the algorithm where the heal technique is implemented. Remaining part of the population that is not healed is important because it:

- ensures functionality of the GA in case that heal technique has no effect due to improper variables-constraints connection,
- provides diversity in the population if the healing was unable to produce feasible designs, so it can be applied again in the following iteration,
- can be useful for the later stages of the optimization, after finding the feasible design.

Crossing-over two designs can change them radically. Therefore, this operator is disabled when using the heal technique in order to increase the success of producing a feasible design. For the part of the population which is not healed, the crossover is used in a standard way to ensure normal GA process. Mutation operator is used on all designs without any exception as it changes the designs far less than the cross-over and does not present a big threat to their feasibility.

Heal technique is intended to create feasible designs in the first attempt, but even if it is unsuccessful, it should considerably reduce the overall constraint violation of the designs, positioning them close to feasible space. It can be used iteratively until feasible designs are found. Thereafter, it is replaced with the repair procedure.

2.2 Repairing

The repair procedure can be applied after obtaining the variable set that yields the feasible solution. If there are several feasible alternatives, the one with the best objective(s) value(s) is taken as a reference. It can be used as a model to repair the infeasible variables from the infeasible solutions, relying on the same variable-constraint mapping that is used for the healing procedure.

Which designs will be repaired depends upon their performance in objective space that is determined by Eq. (3). Only those infeasible designs that are better in objective(s) value(s) than the reference design will be repaired to make them feasible, simultaneously trying to avoid severe performance deterioration. Obviously there is no point in repairing an infeasible solution which originally performs worse than the reference design. Thus one can apply the repair technique on N_{REP} number of infeasible designs in the population which is automatically reduced if there is not as many better than the reference.

Assuming the existence of infeasible designs in the population which are performing better than the best feasible design, the following repairing technique can be applied to generate improved designs:

Step 1: Rank the population according to the performance of designs in objective(s) space by using Eq. (3).

Step 2: Determine the value of d_{OBJ} for the best feasible design and mark it as a reference design.

Step 3: Check how many infeasible designs are better than the reference design. If there is less than N_{REP} of them, assign that number to $N_{REP,REAL}$, otherwise $N_{REP,REAL} = N_{REP}$.

Step 4: To the best $N_{REP,REAL}$ infeasible solutions identify infeasible variables.

Step 5: Starting with the best infeasible design, do for each $N_{REP,REAL}$: keep all feasible variables and replace the infeasible with the corresponding reference design's variables.

Remaining $N - N_{REP,REAL}$ individuals are selected and processed following the normal procedure of the used optimizer. Variables from those designs that are repaired get mutated, but cross-over is disabled to prevent any possible performance degradation.

Repair technique can be used repeatedly with each new population, if there is at least one feasible solution present. If there is none, the heal technique is re-activated to create it. Healing the designs in such situation is much easier than in the beginning when they are far from the feasible space. Now only few variables are infeasible, so there is far greater chance of correcting them successfully. Healed designs should not differ much in objective(s) value(s) from the original. So throughout the optimization heal and repair strategies are active to aid in more efficient optimization.

3 'VOP' GENETIC ALGORITHM

To implement heal and repair strategies (H&R) a simple GA called VOP is used (Klanac and Jelovica 2007). VOP is a binary coded algorithm consisting of: a) a fitness calculator, b) the weighted roulette wheel selector operating on the basis of computed fitness values, and c) a subroutine executing the single-point cross-over with a probability of p_C and the bit-wise mutation with a probability of p_M. These are standard operators and are, except for the fitness calculator, elaborated in Deb (2001).

VOP's fitness function is defined as:

$$\varphi(\boldsymbol{x},i) = \left(\max_{x\in\mathbf{X}^i}\left[d(\boldsymbol{x},i)\right] - d(\boldsymbol{x},i)\right)^{d(\boldsymbol{x},i)} \tag{4}$$

where the distance $d(\boldsymbol{x},i)$ of a design *i* from the origin of normalized objective space is obtained as:

$$d(\boldsymbol{x},i) = \left\{\sum_k \left[\overline{f}_k(\boldsymbol{x},i)\right]^2\right\}^{1/2}, \forall k \in [1, M+J]. \tag{5}$$

and *m* is the number of constraints. Normalization of objective f_k for design *i* is linearly performed using

$$\overline{f}_k(\boldsymbol{x},i) = \frac{f_k - \min\limits_{\forall x\in\mathbf{X}^i} f_k}{\max\limits_{\forall x\in\mathbf{X}^i} f_k - \min\limits_{\forall x\in\mathbf{X}^i} f_k}. \tag{6}$$

4 OPTIMIZATION OF A TANKER STRUCTURE

Optimization will be performed on the 40 000 DWT and 180m long chemical tanker's midship section. Its main frame longitudinal elements are optimized for the smallest allowed scantlings, while keeping the topology of the structure unchanged. The tank's plating is built from duplex steel to resist the aggressive chemicals which are transported and is the only part of the structure with yield strength of 460 MPa. For the remaining structure, 355 MPa steel is used.

4.1 Problem description

Variables considered in the optimization are the plate thicknesses and stiffener sizes. Number of stiffeners per each strake is depicted in Figure 1 together with scantlings of the transversal structure. Considered variable values are discrete, having the step for the plates of 1 mm, a value in general appropriate for early design stage. Plate thicknesses in double bottom and double side

structure are assumed to be available from 8 to 23mm, except of stringers whose range is the same as for longitudinal bulkhead and the deck, 5 to 20mm. Stiffener sizes are taken as standard *holland* profiles.

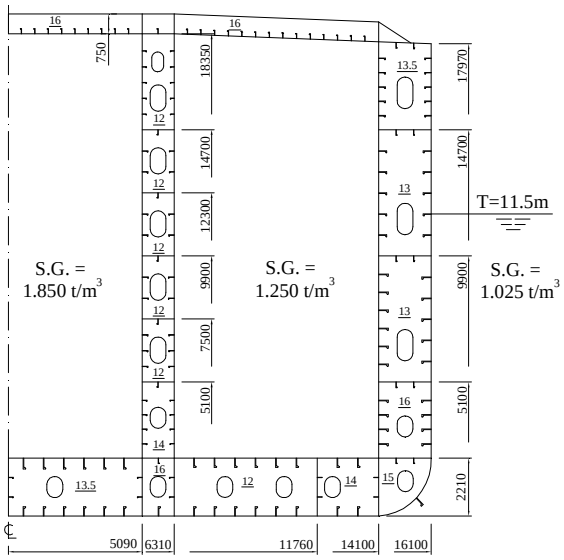

Figure 1. Half of the main frame section of a considered tanker and scantlings of the transverse structures (underlined)

The midship section is assumed to stretch between L/4 and 3L/4 cross-sections, without the change in scantlings. It is subjected to the hull girder loads, the cargo loads and the lateral hydrostatic loads, while ballast tanks are assumed to be empty. Pressure loads are calculated from liquid density indicated in Figure 1. The section is exposed to four critical hull girder loads acting on a section at two positions, L/4 and L/2. For the former, the shear force of 72 960 kN is applied in hogging and -74 880 kN in sagging, while for the later, the total vertical bending moment of 2 933 000 kNm is assumed for hogging and -2 410 000 kNm for sagging.

The response under the hull girder loads is calculated applying the numerical Coupled Beam method of Naar et al. (2004). On top of that is added the response of the panel under the cargo and hydrostatic loads, calculated with uniformly loaded simple beam.

Each strake is checked for eight failure constraints concerning plate yield and buckling, stiffener yield, lateral and torsional buckling, stiffener's web and flange buckling and crossing-over. These criteria are taken from DNV (2005), Hughes (1988) and Hughes et al. (2004). Altogether 376 failure criteria are calculated for each loading condition. They are transformed into adequacies which is a non-linear normalization function between the structural capacity of some structural element j, $a_j(\boldsymbol{x})$, and a loading demand acting on it, $b_j(\boldsymbol{x})$:

$$g_j(\boldsymbol{x}) = \frac{a_j(\boldsymbol{x}) - |b_j(\boldsymbol{x})|}{a_j(\boldsymbol{x}) + |b_j(\boldsymbol{x})|} \tag{7}$$

Two objectives are considered in this study: minimize the total weight of hull steel (abbreviated as f_1) and minimize the total weight of the duplex steel (abbreviated as f_2). Minimizing the hull weight would increase the payload capacity, but reduction in duplex steel would be the most significant for decrease in production costs since the material and labour costs for this steel are tenfold to those of high tensile 355 MPa steel used for the remaining structure.

Both weights are calculated by extending the obtained cross-sectional weight for the whole length of the ship. For the hull steel weight, additional 21.4t each 3.56m is added to account for the weight of the web frames.

5 THE OPTIMIZATION PROCESS

The optimization is carried out with the following parameters: population of 60 design alternatives, crossover probability 0.8 and mutation probability 0.003. Values are set based on the literature (Deb 2001) and previous experiences of the case (Klanac et al. 2008).

Before allowing H&R to operate, a relation between constraints and variables is established: if any of eight constraints from a strake is violated, assume both strake variables to be responsible (plate thickness and stiffener size). This is physically not entirely correct since stiffener-related limitations are not influenced by plate thickness and vice-versa. To satisfy cross-over constraint however it is better to change both variables, so this rule is for simplicity used for all constraints. Changing one part will influence the stress distribution throughout the structure and thus influence all the constraints. Therefore creating a feasible design from the infeasible in the proposed way cannot be guaranteed.

Random initial population is generated that will be used for all the optimization runs. It is completely infeasible, violating from 14 to 31 constraints. This number is actually not that high comparing to total 376 that need to be satisfied for a design to be feasible. Thus their constraints are mostly fulfilled, and those that are not indicate on infeasible variables, following the simple rule for their detection that was set.

5.1 Effects of healing the population

Optimizer starts to operate and determines that the initial population (P-A) is completely infeasible so it activates the healing procedure. Healing is used for 25 designs (41% of population) based on the smallest overall constraint violation, (P-B), 25 based on the best performance, (P-C) and the remaining 10 will be handled as usual with VOP, (P-D). Figure 2 shows the change in number of violated constraints after healing the first generation, according to defined population parts.

Originating from the solutions closest to the x-axis in P-A, first 25 designs from the second generation (P-B) are in average the heaviest designs in that generation as seen in Figure 3 and Figure 4, showing the f_1 and f_2, respectively. Thicker plate or larger stiffener generally cause smaller violation of associated constraints, thus heavier structure is usually less infeasible. That is why designs that are healed based on the smallest constraint violation are heavier (or worse in performance) than those healed based on objectives. The latter are located in P-C in Figure 3 and Figure 4. Table 1 shows that both feasible and infeasible average design in P-B is heavier than in P-C, in both objectives. Figure 3 and Figure 4 visualize also the decrease of feasible solutions in P-C compared to P-B. Contrary to first 25 designs in the healed generation, second 25 originally brake more constraints and are harder to make feasible. Figure 2 and Table 1 confirm this reduced feasibility in P-C. Majority of feasible solutions created using healing is present in the first half of P-B and in the middle and second half of P-C part of the population – those having originally less infeasible variables. Heal technique managed to create totally 18 feasible designs out of 50 attempts (36%), although the remaining designs are also competitive, braking up to four constraints but mainly only one. A qualitative feasible design can arise if those infeasible designs are crossed-over in further stages of optimization.

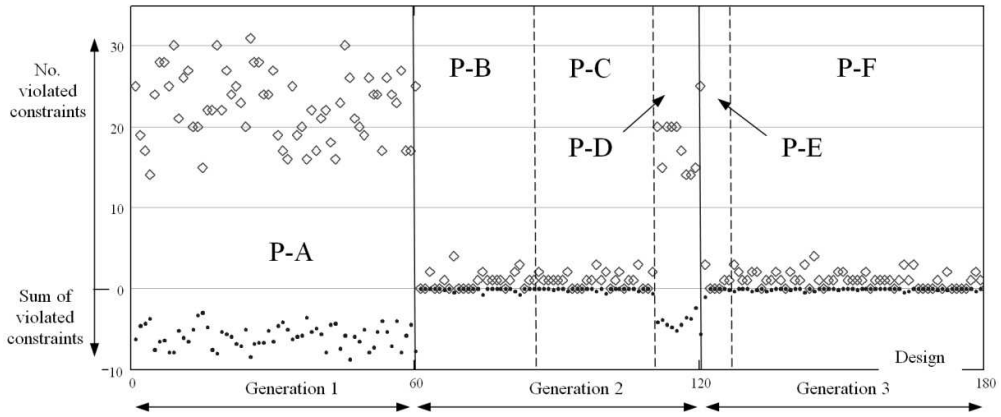

Figure 2. VOP – H&R: Constraint violations of the first 180 designs

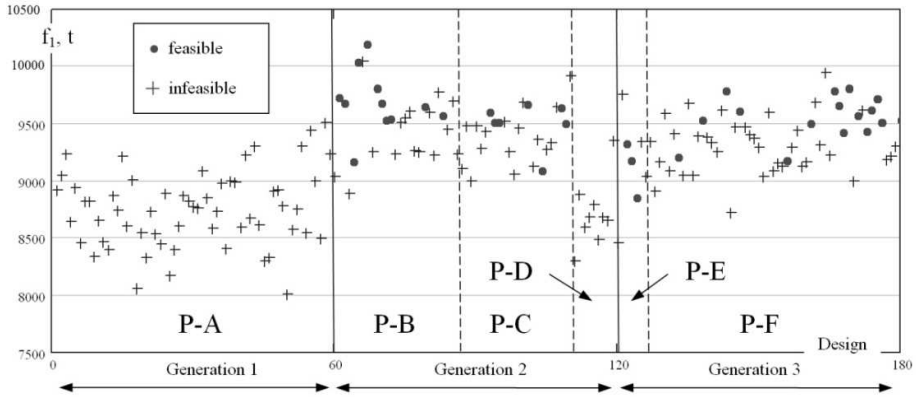

Figure 3. VOP – H&R: Total hull steel weight of the first 180 designs

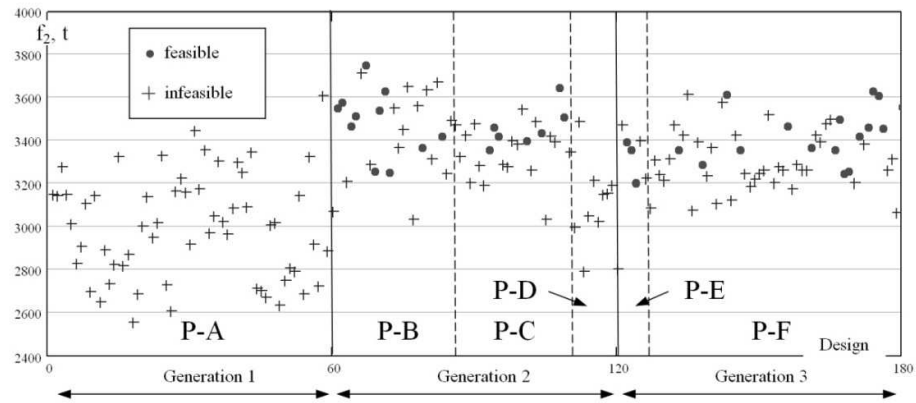

Figure 4. VOP – H&R: Total duplex steel weight of the first 180 designs

Remaining 10 designs for the second generation (P-D) are selected and processed following the normal VOP procedure. They can be seen in Figure 2, Figure 3 and Figure 4. We can see the reduction in number of violated constraints when compared to first generation where they originate

from; see Figure 2. Obviously selection preferred less-infeasible designs having up to 20 broken constraints (and one with 25), which is smaller than the 31 in the initial population.

Table 1. VOP – H&R: Performance of different population parts from the first three generations

Generation		1	2			3	
Population part		P-A	P-B	P-C	P-D	P-E	P-F
Working mode		normal	heal cons	heal objs	normal	repair	normal
Number of designs		60	25	25	10	6	54
Number of feasible designs		0	11	7	0	3	16
Feasible	Avg. hull steel, t	-	9686	9500	-	9113	9550
	Avg. duplex, t	-	3483	3458	-	3317	3431
Infeasible	Avg. hull steel, t	8751	9451	9366	8685	9374	9300
	Avg. duplex, t	2997	3441	3344	3086	3364	3299

Comparing designs in P-D to those in P-B and P-C in Figure 3 and Figure 4 clarifies the reason for not healing the whole second generation – those solutions are much better in objective values. Table 1 shows that average infeasible design from P-D is in fact 800 t lighter in f_1 and 300 t lighter in f_2 than the average design from P-B and P-C. This will provide diversity for the continuation.

5.2 Effects of repairing the population

Continuing with the third generation, infeasible designs from part P-D are repaired since their objective(s) value(s) are better than the best feasible design. The best six of them will undergo repairing (P-E). Six designs or 10% of the population is the maximum amount of individuals that is repaired during the optimization. This will then not interfere much in normal GA operations in case the repair is not effective.

The best feasible solution from the second generation is selected as a 'donor' for repairing. Higher competition among input feasible solutions will yield a better donor design, so it is preferable to have more feasible individuals to choose from. Six designs from the second generation to be repaired in the third are there the best infeasible design, likely from population part P-D. Their infeasible variables are replaced with the feasible ones from the donor design. Repaired designs are presented in P-E in Figure 2, Figure 3 and Figure 4. It can be seen that three of them are repaired successfully while others are almost feasible. The first repaired solution is originally the lightest in the second generation and therefore requires replacing many infeasible variables. After repairing, it stays infeasible and becomes quite heavy. The following three designs are more successful, turning into feasible and importantly, lighter in both objectives than all feasible solutions from the second generation. This is tabulated in Table 1 where it can be seen that the average feasible design in P-E is 4-500 t lighter in f_1 than the average feasible design in P-B and P-C. Weight of duplex steel is some 150 t smaller in the same comparison. These designs will surely be selected in the following generation.

Remaining part of the population from the third generation (P-F) is taken from parts P-B, P-C and P-D and is processed normally. The weight of the resulting designs can be seen in Figure 3 and Figure 4. Clearly among the feasible designs in P-F there are no heavier designs from P-B. Having low fitness, they were not selected for continuation. f_1 of the average feasible design in P-F is 70 t lighter than in combined P-B and P-C population, while the f_2 is reduced by 35 t; see Table 1. Infeasible designs are in P-F also lighter than the ones in P-B and P-C where they originate from. They are not descendants from P-D since designs in P-F violate up to four constraints, much smaller than in P-D.

5.3 Comparing the performance of the algorithms

Optimization with VOP – H&R is continued until generation 350 when the progress became negligible. Starting with the same initial population and using the same parameters, normal VOP and also NSGA-II algorithm was run to compare the results but were both allowed continuing until generation 600. NSGA-II is a widespread multi-objective GA that uses the following concepts: elitism, non-dominated sorting of designs in objective space, crowding-distance operator, binary tournament selection and constraint-domination. Description of these concepts and the algorithm itself can be found in Deb et al. (2002).

Comparison of the optimization progress is tabulated in Table 2. Comparing the objectives in the given generations, clearly VOP - H&R surpasses other two algorithms. This is especially expressed in the beginning since the heal procedure reduced the search for feasible designs to only one generation, for which NSGA-II spend 15 and normal VOP 30 generations. The final design with VOP - H&R is for f_1 better by 90 t and 120 t from VOP and NSGA-II, respectively, while for f_2 by 35 and 100 t in the same algorithm order although the last two algorithms were run almost twice as long.

Table 2. Comparison of optimizers, presenting weight of hull steel (f_1) and weight of duplex steel (f_2) for the best designs of each objective in the given generations

Algorithm	VOP		VOP - H&R		NSGA-II	
Objective	f_1, t	f_2, t	f_1, t	f_2, t	f_1, t	f_2, t
Generation 50	8835	3150	7825	2715	8340	2795
Generation 100	8115	2835	7580	2680	7780	2760
Generation 200	7885	2795	7420	2645	7545	2735
Generation 300	7775	2745	7350	2645	7520	2735
End	7440	2674	7348	2632	7467	2734

5.4 Optimization outcome

Figure 5 presents the best design alternative from the three optimization runs. Its scantlings are not standardized, thus there can be significant differences in plate thicknesses and stiffener sizes from neighboring strakes. This design is compared with design of *minimum rule scantlings* obtained from the calculation of the rules of Bureau Veritas (2006). There is no corrosion addition in any of the presented designs.

In this study high loads are imposed to act on the main frame, requiring that it is not only used in L/2, where it has to deal with maximum vertical bending moment, but also in L/4 and 3L/4 where it is subdue to maximum shear force. Due to this conservative consideration, the minimal rule scantlings design shown in Figure 5, violates 32 constraints, mostly plate and stiffener yielding from the vertical strakes in longitudinal bulkhead, where it does not account for high shear force, but also plate yield in inner bottom. To account for this, plates are in inner bottom and also in bottom strakes for the optimized structuresignificantly thickened, however corresponding stiffeners are reduced guided by weight reduction. The optimized structure also has considerably higher plate thicknesses but also bigger stiffeners in longitudinal bulkhead. This is especially expressed for the lowest strakes, who deal with high cargo pressure. Those plates are the main reason for the increase of the weight of duplex steel, as seen in Table 3. Outer strakes in double side have generally smaller scantlings, while the inner are similar to bottom strakes – thickened plates and reduced stiffeners. Deck scantlings are reduced while the reduction in all stringers is quite significant.

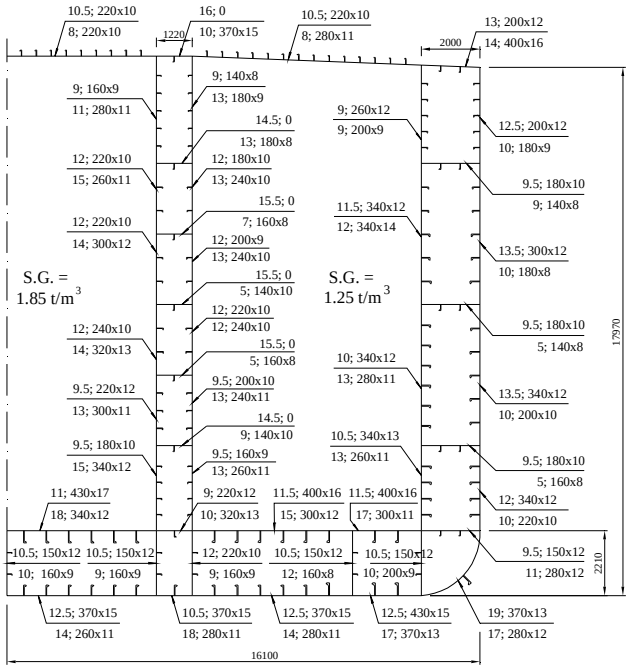

Figure 5. Scantlings of the main-frame members for the design of minimal rule requirements, shown above dimension lines, and for the design $\mathbf{x}H^{**}$, shown below dimension lines

The optimized structure has approached many constraint boundaries, mainly plate yield, plate buckling and stiffener yield from majority of strakes in the main frame, except for stringers in double side and cofferdam but also floors, whose constraint values are generally increased. That design has in total 52 active constraints, as seen in Table 3, where constraint is considered to be active if stress exceeds 3/4 of critical value.

Table 3. Objective's values for the best hull steel design obtained in optimization, compared with design of minimal rule requirements, and also the number of negative and active constraints

	Optimized structure	Min. rule req.
f_1, t	7353	7550
f_2, t	2652	2270
Negative con.	0	32
Active con.	52	18

6 CONCLUSION

The 'healing' and 'repairing' techniques for the genetic algorithm can reduce the number of evaluated designs needed to reach satisfactory optimization solutions. This allows resource and time savings when used in the early design stage. Benefits of these techniques have been shown on example of the main frame of 40 000 DWT chemical tanker. Optimization time was reduced by approximately half when compared to the 'normal' working mode of the algorithm and also when compared to one well-established GA. Crucial detail was to establish proper connection between the constraints and the variables. This was done here in a rather crude manner, however, even with that kind of mapping the results showed clear advantages of using healing and repairing techniques.

In general, it can be assumed that proper sensitivity between variables and constraints ensures the success of creating feasible designs. It also presumably allows capturing the good objective(s) values(s) in the later stages of the optimization when the repair technique is utilized. These assumptions require more thorough investigation.

Objectives used in this study do not form well-developed Pareto frontier, thus the designs are spaced rather condense in the objective space and it is easy to rank them there. The question arises on the proper selection scheme when dealing with large and dense frontier. This is definitely one of the questions to be answered in the following studies.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of IMPROVE project, funded by European Union (Contract nr. 031382- FP6 2005 Transport-4), and the Technology Development Centre of Finland – TEKES, including Finnish shipbuilding industry, through the project CONSTRUCT. First author would also like to acknowledge the support of Finnish National Graduate School of Engineering Mechanics.

REFERENCES

Bureau Veritas 2006. *Rules for the classification of steel ships.*

Cheong, F., Lai, R. 2000. *Constraining the Optimization of a Fuzzy Logic Controller Using an Enhanced Genetic Algorithm*, IEEE Trans on Systems, Man, and Cybernetics – part B: Cybernetics, 30(1), p. 31-46.

Chou, H., Premkumar, G., Chu, C.H. 2001. *Genetic algorithms for Communications Network Design – An Empirical Study of the Factors that Influence Performance*, IEEE Trans on Evol Comp, 5(3), p. 236-249.

Deb, K. 2001. *Multi-Objective Optimization Using Evolutionary Algorithms*. Chichester: John Wiley & Sons.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. 2002. *A Fast and Elitist Multiobjective Genetic Algorithm* – NSGA-II, IEEE Transactions on Evolutionary Computation. 6/1: 182-197.

Deb, K., Mohan, M., Mishra, S. 2003. *A Fast Multi-objective Evolutionary Algorithm for Finding Well-Spread Pareto-Optimal Solutions*, KanGAL Report Number 2003002

Det Norske Veritas 2005. *Rules for the classification of steel ships*. Høvik.

Ehlers, S., Klanac, A., Tabri, K. 2007. *Increased safety of a tanker and a RO-PAX vessel by implementing a novel sandwich structure*. 4th Int. Conference on Collision and Grounding of Ships. Hamburg:109-115.

Hughes, O.F. 1988. *Ship Structural Design*. Society of Naval Architects and Marine Engineers. New York:Wiley.

Hughes, O.F., Ghosh, B., Chen, Y. 2004. *Improved prediction of Simultaneous local and overall buckling of stiffened panels*. Thinn-Walled Structures, 42: 827-856.

Klanac, A., Jelovica, J. 2009. *Vectorization and Constraint Grouping to Enhance Optimization of Marine Structures.* Marine Structures, 22(2): 225-245.

Klanac, A., Ehlers, S., Jelovica, J. 2009. *Optimization of crashworthy marine structures*. In press by Marine Structures, available online 12 August 2009

Liu, B., Haftka, R.T., Akgün, M.A., Todoroki, A. 2000. *Permutation genetic algorithm for stacking sequence design of composite laminates*, Comp. methods in applied mechanics and engineering, 186, p. 357-372.

Naar, H., Varsta, P., Kujala, P. 2004. *A theory of coupled beams for strength assessment of passenger ships*, Marine Structures, 17(8): 590-611.

Osyczka, A. 2002. *Evolutionary Algorithms for Single and Multicriteria Design Optimization*. New York: Physica-Verlag.

Romanoff, J., Klanac, A. 2007. *Design Optimization of a Steel Sandwich Hoistable Car-Decks Applying Homogenized Plate Theory*. 10th Int. Symp. on Practical Design of Ships and Other Floating Structures – PRADS, Houston.

Todoroki, A., Haftka, R.T. 1998. *Stacking sequence optimization by a genetic algorithm with a new recessive gene like repair strategy*, Composites Part B, Elsevier Science, p. 277-285.